PARALLEL ALGOL: Combining Procedures with Concurrency

Stephen Brookes

Department of Computer Science
Carnegie Mellon University

ESSENTIALS

- Parallel Algol =

  shared-variable parallel programs
  + call-by-name λ-calculus

- simply typed

  $$\theta ::= \mathbf{exp}[\tau] \mid \mathbf{var}[\tau] \mid \mathbf{comm} \mid (\theta \to \theta') \mid \theta \times \theta'$$

  phrase types

  $$\tau ::= \mathbf{int} \mid \mathbf{bool}$$

  data types

- recursion and conditional at each type

cf. Reynolds: *The essence of* Algol

RATIONALE

- Programs can cooperate by reading and writing shared memory
- Procedures can encapsulate parallel idioms (e.g. mutual exclusion)
- Local variable declarations can be used to limit the scope of interference

INTUITION

Procedures and parallelism should be *orthogonal*:

- combine smoothly
- "modular" semantics
- conservative extension

MUTUAL EXCLUSION

```
procedure mutex(n1, c1, n2, c2);
boolean s;
begin
        s:=true;
        while true do
           (n1;  await s  then s:=false;
            c1;  s:=true)
     || while true do
           (n2;  await s  then s:=false;
            c2;  s:=true)
end
```

- Encapsulates common use of a *semaphore*
- Correctness relies on *locality* of s
- Independent of n_i and c_i

OUTLINE of SEMANTICS

- Traditional "global state" models fail to validate natural equivalences, e.g.

$$\mathbf{new}[\tau]\ \iota\ \mathbf{in}\ P = P$$

  when ι does not occur free in P.

- Need to distinguish between global and local entities

- We adapt "possible worlds" model of ALGOL to the parallel setting...

- ...and extend our "transition trace" semantics (LICS'93) to include procedures and recursion.

- We adapt a "parametric" model of ALGOL to the parallel setting...

- ...and introduce a form of relational reasoning for shared-variable programs.

POSSIBLE WORLDS

- The shape of the state changes as program runs
- A “possible world” W represents the set of currently allowed states
- For sequential Algol, a command denotes a suitable state transformer

$$[\![\mathbf{comm}]\!]W = W \to W_\perp$$

- The meaning of c varies “uniformly” across worlds

$$[\![c_0; c_1]\!]Wu = [\![c_1]\!]Wu \circ [\![c_0]\!]Wu$$

Reynolds, Oles

CATEGORY of WORLDS

- Objects are countable sets
- Morphisms are "expansions":

$$h = (f, Q) : W \to X$$

  - f is a function from X to W
  - Q is an equivalence relation on X
  - f puts each Q-class in bijection with W

INTUITION

- X is a set of "large" states extending the "small" states of W
- f extracts the "small" part of a state
- Q equates states with the same extra parts

Oles

EXPANSIONS

- For each pair of objects W and V there is a canonical *expansion*

$$- \times V : W \to W \times V$$

given by

$$- \times V = (\mathrm{fst} : W \times V \to W,\ Q)$$

where

$$((w_0, v_0), (w_1, v_1)) \in Q \iff v_0 = v_1$$

- Up to isomorphism, every morphism is like this.

INTUITION

$- \times V_\tau$ models the introduction of a local variable of datatype τ.

SEMANTICS

- Types denote functors from worlds to domains, $[\![\theta]\!] : \mathbf{W} \rightarrow \mathbf{D}$
- Type environments denote functors
- Phrases denote natural transformations

  $$[\![P]\!] : [\![\pi]\!] \stackrel{.}{\rightarrow} [\![\theta]\!]$$

  i.e. when $h : W \rightarrow X$,

  $$\begin{array}{ccc} [\![\pi]\!]W & \xrightarrow{[\![P]\!]W} & [\![\theta]\!]W \\ {\scriptstyle [\![\pi]\!]h}\downarrow & & \downarrow{\scriptstyle [\![\theta]\!]h} \\ [\![\pi]\!]X & \xrightarrow[{[\![P]\!]X}]{} & [\![\theta]\!]X \end{array}$$

  commutes.

Naturality enforces locality

CARTESIAN CLOSURE

- The functor category $\mathbf{D}^{\mathbf{W}}$ is cartesian closed.
- Use ccc structure to interpret arrow and product types

$$[\![\theta \times \theta']\!] = [\![\theta]\!] \times [\![\theta']\!]$$
$$[\![\theta \to \theta']\!] = [\![\theta]\!] \Rightarrow [\![\theta']\!]$$

- Thus, procedures will be natural and respect locality.

PROCEDURES

Procedures of type $\theta \rightarrow \theta'$ denote, at world W, natural families of functions $p(-)$: if $h : W \rightarrow X$ and $h' : X \rightarrow Y$,

$$\begin{array}{ccc} [\![\theta]\!]X & \xrightarrow{\quad p(h) \quad} & [\![\theta']\!]X \\ {\scriptstyle [\![\theta]\!]h'}\Big\downarrow & & \Big\downarrow{\scriptstyle [\![\theta']\!]h'} \\ [\![\theta]\!]Y & \xrightarrow[\quad p(h;h') \quad]{} & [\![\theta']\!]Y \end{array}$$

commutes.

Procedures can be called at expanded worlds, and naturality enforces locality constraints.

COMMANDS

- Commands denote *closed* sets of *traces*:

$$[\![\mathbf{comm}]\!]W = \wp^{\dagger}((W \times W)^{\infty})$$

- Trace sets are *closed under stutters*

$$\alpha\beta \in c \;\&\; w \in W \;\Rightarrow\; \alpha(w, w)\beta \in c$$

  and *closed under mumbles*

$$\alpha(w, w')(w', w'')\beta \in c \;\Rightarrow\; \alpha(w, w'')\beta \in c$$

- $[\![\mathbf{comm}]\!]h$ converts a trace set over W to a trace set over X:

$$[\![\mathbf{comm}]\!](f, Q)c = \{\beta \mid \mathrm{map}(f \times f)\beta \in c \;\&\; \mathrm{map}(Q)\beta\}$$

INTUITION

- A trace

  $$(w_0, w'_0)(w_1, w'_1) \ldots (w_n, w'_n) \ldots$$

  represents a fair interactive computation.

- Each step (w_i, w'_i) represents a finite sequence of atomic actions.

- $[\![\mathbf{comm}]\!]hc$ behaves like c on the W-component of state, has no effect elsewhere.

EXPRESSIONS

Expressions denote trace sets:

$$[\![\mathbf{exp}[\tau]]\!]W = \wp^{\dagger}(W^{+} \times V_{\tau} \ \cup \ W^{\omega})$$

$$\begin{aligned}[\![\mathbf{exp}[\tau]]\!](f, Q)e = \{(\rho', v) \mid (\mathrm{map} f \rho', v) \in e\} \\ \cup \{\rho' \mid \mathrm{map} f \rho' \in e \cap W^{\omega}\}\end{aligned}$$

VARIABLES

"Object-oriented" interpretation:

variable = acceptor + expression

$$[\![\mathbf{var}[\tau]]\!]W = (V_{\tau} \to [\![\mathbf{comm}]\!]W) \times [\![\mathbf{exp}[\tau]]\!]W$$

skip

Finite stuttering:

$$
\begin{aligned}
[\![\mathbf{skip}]\!]Wu &= \{(w,w) \mid w \in W\}^\dagger \\
&= \{(w,w) \mid w \in W\}^+
\end{aligned}
$$

Never changes the state,
always terminates

ASSIGNMENT

Non-atomic; source evaluated first:

$$
\begin{aligned}
&[\![I{:=}E]\!]Wu = \\
&\quad \{(\mathrm{map}\Delta_W \rho)\beta \mid (\rho, v) \in [\![E]\!]Wu \\
&\qquad\qquad \& \ \beta \in \mathrm{fst}([\![I]\!]Wu)v\}^\dagger \\
&\quad \cup \{\mathrm{map}\Delta_W \rho \mid \rho \in [\![E]\!]Wu \cap W^\omega\}^\dagger.
\end{aligned}
$$

PARALLEL COMPOSITION

Fair merging of traces:

$$
\begin{aligned}
&[\![P_1 \| P_2]\!] W u = \\
&\qquad \{\alpha \mid \exists \alpha_1 \in [\![P_1]\!] W u,\ \alpha_2 \in [\![P_2]\!] W u. \\
&\qquad\qquad (\alpha_1, \alpha_2, \alpha) \in \mathit{fairmerge}_{W \times W}\}^{\dagger}
\end{aligned}
$$

where

$$
\begin{aligned}
&\mathit{fairmerge}_A = \mathit{both}_A^{*} \cdot \mathit{one}_A \cup \mathit{both}_A^{\omega} \\
&\mathit{both}_A = \{(\alpha, \beta, \alpha\beta), (\alpha, \beta, \beta\alpha) \mid \alpha, \beta \in A^{+}\} \\
&\mathit{one}_A = \{(\alpha, \epsilon, \alpha), (\epsilon, \alpha, \alpha) \mid \alpha \in A^{\infty}\}
\end{aligned}
$$

This is natural!

LOCAL VARIABLES

$$
\begin{aligned}
&[\![\mathbf{new}[\tau]\ \iota\ \mathbf{in}\ P]\!] W u = \{\mathrm{map}(\mathrm{fst} \times \mathrm{fst})\alpha \mid \\
&\quad \mathrm{map}(\mathrm{snd} \times \mathrm{snd})\alpha \text{ interference-free } \& \\
&\quad \alpha \in [\![P]\!](W \times V_\tau)([\![\pi]\!](- \times V_\tau)u \mid \iota : (a, e))\}
\end{aligned}
$$

- No external changes to local variable
- $(a, e) \in [\![\mathbf{var}[\tau]]\!](W \times V_\tau)$ is a "fresh variable" representing the V_τ part of the state

AWAIT

$$
\begin{aligned}
&[\![\textbf{await } B \textbf{ then } P]\!]Wu = \\
&\qquad \{(w, w') \in [\![P]\!]Wu \mid (w, \mathtt{tt}) \in [\![B]\!]Wu\}^{\dagger} \\
&\quad \cup \{(w, w) \mid (w, \mathtt{ff}) \in [\![B]\!]Wu\}^{\omega} \\
&\quad \cup \{\mathrm{map}\Delta_W \rho \mid \rho \in [\![B]\!]Wu \cap W^{\omega}\}^{\dagger}.
\end{aligned}
$$

- P is atomic, enabled when B is true.
- Busy wait when B is false.

λ-CALCULUS

$$[\![\iota]\!]Wu = u\iota$$

$$[\![\lambda\iota : \theta.P]\!]Wuha = [\![P]\!]W'([\![\pi]\!]hu \mid \iota : a)$$

$$[\![P(Q)]\!]Wu = [\![P]\!]Wu(\mathrm{id}_W)([\![Q]\!]Wu)$$

- This is the standard interpretation, based on the ccc structure.

RECURSION

Requires a careful use of *greatest fixed points*:

- Embed $[\![\theta]\!]W$ in a complete lattice $[\theta]W$ (like $[\![\theta]\!]W$ but without closure and naturality)
- Generalize semantic definitions to $[P]W$.
- Introduce natural transformations

  $$\mathrm{stut}_\theta : [\theta] \stackrel{\cdot}{\longrightarrow} [\theta] \qquad \mathrm{clos}_\theta : [\theta] \stackrel{\cdot}{\longrightarrow} [\![\theta]\!]$$

- Can then define $[\![\mathbf{rec}\ \iota.P]\!]Wu$ to be

  $$\mathrm{clos}_\theta W(\nu x.\mathrm{stut}_\theta W([P]W(u \mid \iota : x)))$$

EXAMPLE

- Divergence = infinite stuttering:

  $$\begin{aligned}[\![\mathbf{rec}\ \iota.\iota]\!]Wu &= (\nu c.\{(w,w)\alpha \mid \alpha \in c\})^\dagger \\ &= \{(w,w) \mid w \in W\}^\omega\end{aligned}$$

LAWS

- This semantics validates:

$$\mathbf{new}[\tau]\ \iota\ \mathbf{in}\ P' = P'$$
$$\mathbf{new}[\tau]\ \iota\ \mathbf{in}\ (P \| P') = (\mathbf{new}[\tau]\ \iota\ \mathbf{in}\ P) \| P'$$
$$\mathbf{new}[\tau]\ \iota\ \mathbf{in}\ (P; P') = (\mathbf{new}[\tau]\ \iota\ \mathbf{in}\ P); P'$$

  when ι does not occur free in P'

- Also (still) validates:

$$(\lambda\iota : \theta.P)(Q) = P[Q/\iota]$$
$$\mathbf{rec}\ \iota.P = P[\mathbf{rec}\ \iota.P/\iota]$$

Orthogonal combination of laws of shared-variable programming with laws of λ-calculus

PROBLEM

Semantics fails to validate

$$\mathbf{new}[\mathbf{int}]\ \iota{:=}0\ \mathbf{in}\ P(\iota{:=}\iota + 1) = P(\mathbf{skip})$$

where P is a free identifier of suitable type

REASON

- Equivalence proof relies on relational reasoning.
- Naturality does not enforce enough constraints on procedure meanings.

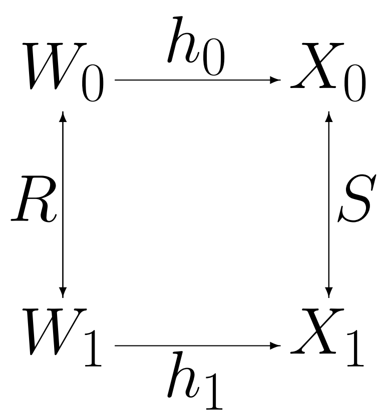
SOLUTION

- Develop a parametric semantics. . .

O'Hearn and Tennent

RELATIONS

- Category of relations $R : W_0 \leftrightarrow W_1$
- A morphism from R to S is a pair (h_0, h_1) of morphisms in $\mathbf{W}$ such that



i.e. (h_0, h_1) respects R and S.

PARAMETRICITY

- Types denote *parametric* functors
  - $[\![\theta]\!]R : [\![\theta]\!]W_0 \leftrightarrow [\![\theta]\!]W_1$
  - $[\![\theta]\!]\Delta_W = \Delta_{[\![\theta]\!]W}$
  - $\forall (d_0, d_1) \in [\![\theta]\!]R.$
    $([\![\theta]\!]h_0 d_0, [\![\theta]\!]h_1 d_1) \in [\![\theta]\!]S$
- Phrases denote *parametric* natural transformations:
  - $\forall (u_0, u_1) \in [\![\pi]\!]R.$
    $([\![P]\!]W_0 u_0, [\![P]\!]W_1 u_1) \in [\![\theta]\!]R$
- The *parametric functor* category is cartesian closed.

COMMANDS

When $R : W_0 \leftrightarrow W_1$ define

$$
\begin{array}{l}
(c_0, c_1) \in [\![\mathbf{comm}]\!] R \iff \\
\quad \forall (\rho_0, \rho_1) \in \mathrm{map}(R). \\
\qquad [\forall \alpha_0 \in c_0.\ \mathrm{map\ fst}\ \alpha_0 = \rho_0 \Rightarrow \\
\qquad\ \exists \alpha_1 \in c_1.\ \mathrm{map\ fst}\ \alpha_1 = \rho_1\ \& \\
\qquad\quad (\mathrm{map\ snd}\ \alpha_0,\ \mathrm{map\ snd}\ \alpha_1) \in \mathrm{map}(R)] \\
\quad \& \\
\qquad [\forall \alpha_1 \in c_1.\ \mathrm{map\ fst}\ \alpha_1 = \rho_1 \Rightarrow \\
\qquad\ \exists \alpha_0 \in c_0.\ \mathrm{map\ fst}\ \alpha_0 = \rho_0\ \& \\
\qquad\quad (\mathrm{map\ snd}\ \alpha_0, \mathrm{map\ snd}\ \alpha_1) \in \mathrm{map}(R)].
\end{array}
$$

This is parametric!

INTUITION

When related commands are started and interrupted in related states their responses are related.

LAWS

- As before,

$$\mathbf{new}[\tau]\ \iota\ \mathbf{in}\ P' = P'$$
$$\mathbf{new}[\tau]\ \iota\ \mathbf{in}\ (P \| P') = (\mathbf{new}[\tau]\ \iota\ \mathbf{in}\ P) \| P'$$
$$\mathbf{new}[\tau]\ \iota\ \mathbf{in}\ (P; P') = (\mathbf{new}[\tau]\ \iota\ \mathbf{in}\ P); P'$$

  when ι does not occur free in P'.

- As before,

$$(\lambda\iota : \theta.P)Q = [Q/\iota]P$$
$$\mathbf{rec}\ \iota.P = [\mathbf{rec}\ \iota.P/\iota]P$$

- In addition,

$$\mathbf{new}[\mathbf{int}]\ \iota{:=}1\ \mathbf{in}\ P(\iota)\ =\ P(1)$$
$$\mathbf{new}[\mathbf{int}]\ \iota{:=}0\ \mathbf{in}\ P(\iota{:=}\iota + 1)\ =\ P(\mathbf{skip}),$$

  relying crucially on parametricity.

EXAMPLE

The programs

new[**int**] $x{:=}0$ **in**
  $(P(x{:=}x+1; x{:=}x+1);$
  if $even(x)$ **then diverge else skip**)

and

new[**int**] $x{:=}0$ **in**
  $(P(x{:=}x+2);$
  if $even(x)$ **then diverge else skip**)

are equivalent in sequential ALGOL but not in PARALLEL ALGOL.

The relation

$$(w,(w',z)) \in R \iff w = w' \,\&\, even(z)$$

works for sequential model but not for parallel.

BOUNDED SEMAPHORES

The phrases

$$\begin{array}{l}\textbf{new}[\textbf{int}]\ x{:=}0\ \textbf{in}\\ \quad P(\textbf{await}\ x < n\ \textbf{then}\ x{:=}x+1,\\ \qquad x{:=}x-1)\end{array}$$

and

$$\begin{array}{l}\textbf{new}[\textbf{int}]\ x{:=}0\ \textbf{in}\\ \quad P(\textbf{await}\ x > -n\ \textbf{then}\ x{:=}x-1,\\ \qquad x{:=}x+1)\end{array}$$

are equivalent in sequential Algol *and* in Parallel Algol.

COUNTERS

The phrases

$$\mathbf{new}[\mathbf{int}]\ x{:=}0\ \mathbf{in}$$
$$P(x{:=}x+1,\ \mathbf{return}(x))$$

and

$$\mathbf{new}[\mathbf{int}]\ x{:=}0\ \mathbf{in}$$
$$P(x{:=}x-1,\ \mathbf{return}(-x))$$

are equivalent in PARALLEL ALGOL.

MORE COUNTERS

The phrases

$$\begin{array}{l}\textbf{new}[\textbf{int}]\ x{:=}0\ \textbf{in}\\ P(x{:=}x+2,\\ \quad \textbf{return}\ (x/2))\end{array}$$

and

$$\begin{array}{l}\textbf{new}[\textbf{int}]\ x{:=}0\ \textbf{in}\\ P(x{:=}x+1; x{:=}x+1,\\ \quad \textbf{return}\ (x/2))\end{array}$$

are *not* equivalent.

COUNTEREXAMPLE

$$P = \lambda(inc,\ val).(inc \| inc;\ val)$$

CONCLUSIONS

- Can blend parallelism and procedures smoothly:
  * faithful to the essence of ALGOL
  * formalizes parallel idioms
  * retains laws of component languages
  * supports relational reasoning, e.g. representation independence
- Semantics by "modular" combination:
  * traces + possible worlds
  * traces + relational parametricity

PRO and CON

- **Advantages**
  - ∗ full abstraction at ground types:
    - ∘ validates natural equivalences
  - ∗ supports common methodology:
    - ∘ object-oriented style
    - ∘ global invariants
    - ∘ assumption–commitment
- **Limitations**
  - ∗ doesn't model irreversibility of state change
  - ∗ not fully abstract at higher types

. . . to be continued?